REMARKS

Claims 1-8, 14-19, 31, and 32 are pending. All pending claims were rejected by the Examiner in the Final Office Action dated January 26, 2009.

Claims 31 and 32 were rejected as being indefinite. They are amended to clarify the intended scope of the claims. No new matter has been added.

Claim Rejections Under 35 U.S.C. §103

Claims 1-8, and 31 are rejected under 35 U.S.C. §103(a) as being unpatentable over Intel, Inc. (IA-32® Architecture Software developer's Manual, Volumes 1-2, 2002) ("Intel"), in view of U.S. Patent No. 5,420,992 to Killian ("Killian"). Applicants respectfully traverse.

Applicants respectfully submit that Intel does not disclose or suggest "wherein substantially all multi-byte aligned branch instructions are operable to access the instructions at byte aligned addresses." In the Office Action, the Examiner points to instruction "JMP-jump" and associated text as disclosing this limitation. Particular, the Office Action states:

> see Intel, Vol. 2, page 3-357 "JMP-Jump" instruction reference; page 3-358, line 1-2, "A relative offset (rel8, re116, or rel32) is generally specified as a label in assembly code, but at the machine code level, it is encoded as a signed 8-, 16-, or 32-bit immediate value."; Examiner's note: In the description of operating modes, Intel discloses a jump instruction that uses an offset corresponding to 8 bits (JMP rel8) as well as other indexing modes (rel16, rel32 et al.).

It appears that the Examiner construes the Intel description to mean that the 8-bit offset allows jumping by one byte. Applicants respectfully disagree. The JMP rel8 instruction is described as "Jump short, relative, displacement relative to next instruction." (Intel Vol. 2, 3-357) A short jump is further described as "a near jump where the jump range is limited to –128 to +127 from the current EIP value." *Id.* The jump range includes 256 possibilities, corresponding to the 8-bit offset size. This EIP is used throughout the JMP operation examples as the instruction address. The EIP value is also defined in the Intel Architecture book:

> The instruction pointer (EIP) register contains the offset in the current code segment for the next instruction to be executed. It is advanced from

> one instruction boundary to the next in straightline code or it is moved ahead or backwards by a number of instructions when executing JMP, Jcc, CALL, RET, and IRET instructions. (Intel Vol. 1, section 3.5, page 3-6)

Accordingly, the EIP offset and therefore the rel8 offset value denotes the number of instructions "ahead or backwards" and not a number of bytes or bits. Because all instructions are multiple bytes (16 bit or 32 bit), the EIP offsets by a number of instructions are necessarily multiple bytes. Thus, Intel does not disclose or suggest "wherein substantially all multi-byte aligned branch instructions are operable to access the instructions at byte aligned addresses."

Further, it is respectfully asserted that Intel, alone or in combination with Killian, fails to teach or render obvious the claim 1 limitation of "common subcircuitry operable to perform sign extensions of an immediate field in non-branch instructions and to perform sign extensions of said immediate field in branch instructions to calculate a target address for branch instructions, wherein said common subcircuitry operating on said non-branch instructions is the same subcircuitry operating upon said branch instructions."

While the secondary reference Killian does disclose a circuit for performing sign extension, it does not teach or disclose a circuit for performing sign extension that "supports multibyte branch extensions operable to access other instructions at byte aligned addresses" as recited in the claim. As discussed above, neither does the Intel reference. This is a nonobvious and nontrivial improvement as noted in the present application in at least the last two paragraphs of the description for Figure 3, reproduced below.

> Although both the branch instruction and the add integer instruction perform sign extension operations on the immediate field value, the branch instruction multiplies the immediate field value by four before performing the sign extension. The multiplication is performed to allow for word alignment and an expanded target branch address range. However, by requiring a multiplication of four before performing the sign extension, different subcircuitry route is required to perform the operation. In many instances, one subcircuit is specifically configured to sign extend an immediate field and another subcircuit is specifically configured to perform a sign extend of an immediate field multiplied by four. In some examples, multiplication by four is performed by using additional multiplexers and shifters on a base sign extended subcircuit. However, having additional hardware or additional subcircuits is expensive,

> particularly for programmable chips. Consequently, and the techniques and mechanisms of the present invention allow the reuse of subcircuitry for both types of instructions.
>
> Techniques and mechanisms of the present invention allow the units of branch instructions to be in bytes even though many of the instructions are multiple bytes in size and a required to be aligned on multi-byte addresses.

Therefore, Killian does not rectify the lack in teachings of Intel because neither reference discloses a circuit that "supports multibyte branch extensions operable to access other instructions at byte aligned addresses." For at least the foregoing reasons, withdrawal of this rejection is respectfully requested.

Claims 14-19 and 32 are rejected under 35 U.S.C. §103(a) as being unpatentable over Intel, Inc. (IA-32® Architecture Software developer's Manual, Volumes 1-2, 2002) ("Intel") in view of U.S. Patent No. 5,420,992 to Killian ("Killian") in view of Wittig et al., "OneChip: An FPGA Processor with Reconfigurable Logic" ("Wittig"). Applicants respectfully traverse.

Claim 14 recites "wherein said common subcircuitry operating on said non-branch instructions is the same subcircuitry operating upon said branch instructions, wherein the sign extension of the immediate field associated with one or more branch instructions is performed to determine a branch target address." The Office Action at page 6 contends that Killian discloses this common subcircuitry.

> Killian teaches a common subcircuitry operating on said non-ranch instructions is the same subcircuitry operating upon said branch instructions wherein the common subcircuitry is configured to perform sign extensions of immediate fields in non-branch instructions and perform sign extensions of immediate fields in branch instructions to calculate a target address for branch instructions (See column 8, lines 7-13, Figure 3C).

Applicants submit that not only does Killian not teach this common subcircuitry, it teaches away from this limitation. The cited passage at column 8, lines 7-13 does not disclose a common subcircuitry. It states:

> In the case of computational (ALU immediate) instructions, the immediate field is extended, combined with the content of the source register, and the result stored in the destination register. In the case of branch instructions

> the immediate field is sign-extended and added to the PC to form a target address.

This passage merely states that for both computational (ALU immediate) instructions and branch instructions, the immediate field is sign extended. This passage makes no suggestion that the same subcircuitry would be used. The Office Action also refers to Figure 3C, reproduced below for convenience. The text associated with Figure 3C makes clear that different subcircuitry is used for sign extension of different instruction types. Column 11, lines 40-50 discloses the organization and address path for a store or load instruction (non-branch):

> Virtual data addresses for load and store instructions are computed by a data address adder 77. Adder 77 combines a base address from one of the registers with an offset derived from the 16-bit immediate field in the instruction. A sign-extension circuit 78 sign-extends the 16-bit offset to 32 bits before the offset is combined at the adder. Depending on whether the instruction is a load or a store, the data entity is read from the addressed memory location and loaded into the destination register, or the data entity in the source register is written into the addressed memory location.

The sign-extension circuit 78, located at the top left of Figure 3C, extends the offset before it is combined at the adder 77. The next paragraph, lines 51-64, discloses the organization and address path for a branch instruction with the branch taken:

> The instruction address portion of the AU includes PC 50 and a next address multiplexer 80 which loads the PC with a selected one of four inputs. . . The third input, selected in the case of a branch instruction with the branch taken, is a branch target address provided by a branch adder 85. The branch adder combines the content of the PC and an offset derived from the 16-bit immediate field in the branch instruction.

In the top right portion of Figure 3C, the offset enters a sign-extension circuit before being combined with the program counter value at the branch adder 85. Separate sign-extension circuits are depicted in the block diagram for different instruction types. Applicants submit that this separate depiction teaches away from a common subcircuitry.

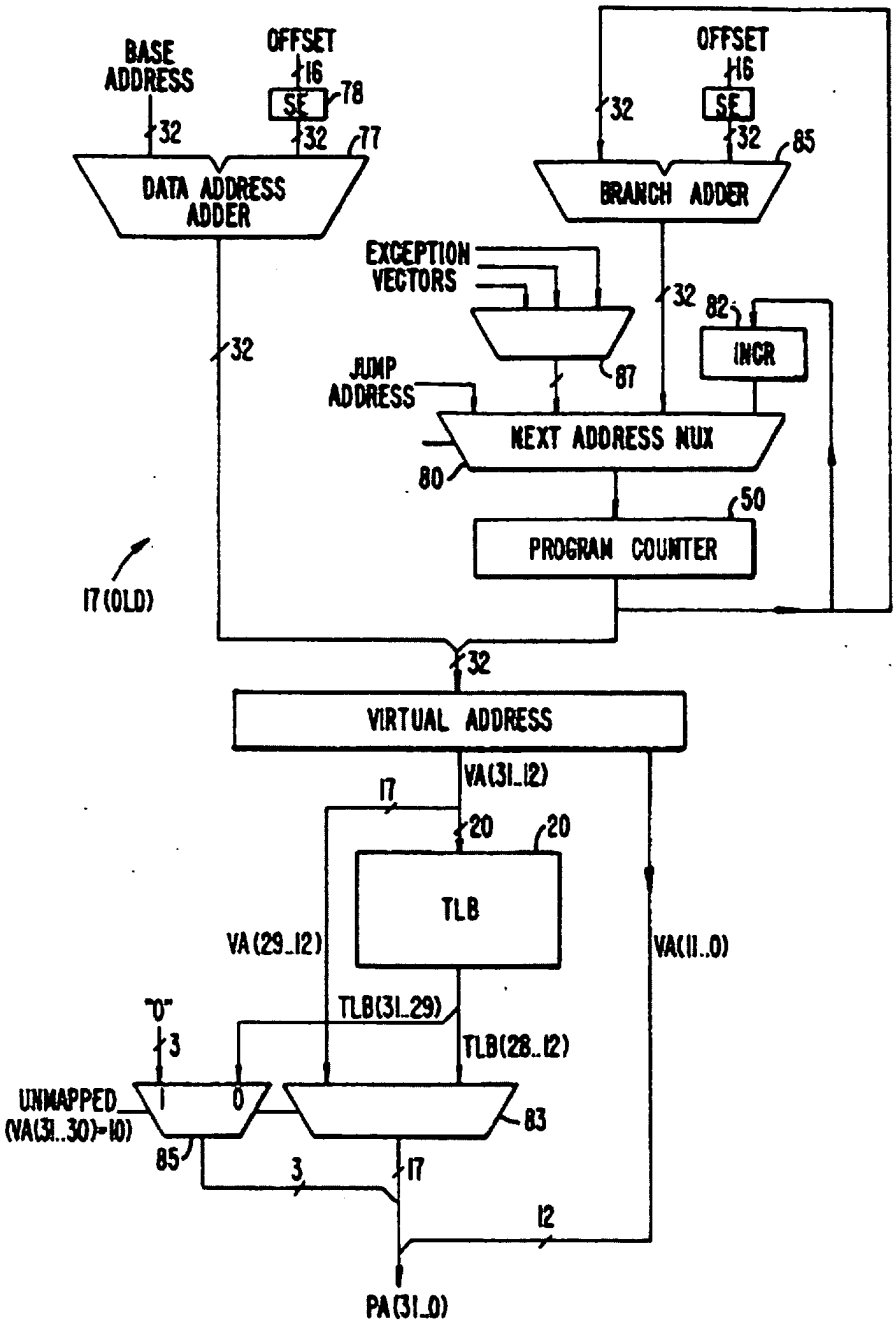

FIG. 3C. (PRIOR ART)

Intel in combination with Killian does not disclose a common subcircuitry as claimed. In fact the Figure 3C and associated text referenced in the Office Action tends to teach away from a common subcircuitry. Wittig does not cure these defects of Intel and Killian. Thus, claim 14

and its dependent claims are not obvious in view of Intel, Killian and Wittig. For at least the foregoing reasons, withdrawal of this rejection is respectfully requested.

With regard to dependent claim 18, it is submitted that Intel, alone or in combination with Killian and Solomon fails to teach the amended limitation "wherein common subcircuitry is used to handle the immediate field associated with the branch and non-branch instructions and wherein an immediate field value is maintained in units of bytes" as discussed above concerning claim 1. Thus, claim 18 is independently patentable over claim 14.

CONCLUSION

Accordingly, it is believed that this application is now in condition for allowance and an early indication of its allowance is solicited. Should the Examiner believe that a telephone conference would expedite the prosecution of this application; the undersigned can be reached at the telephone number set out below.

Please charge any required fees or credit any over payments to Weaver Austin Villeneuve Sampson LLP deposit account 50-4480.

Respectfully submitted,
Weaver Austin Villeneuve & Sampson LLP

/Cindy H. Shu/

Cindy H. Shu
Reg. No. 48,721

P.O. Box 70250
Oakland, CA 94612-0250
510-663-1100